SIAM J. OPTIM.
Vol. 30, No. 1, pp. 980–1006

THE BOOSTED DIFFERENCE OF CONVEX FUNCTIONS ALGORITHM FOR NONSMOOTH FUNCTIONS*

FRANCISCO J. ARAGÓN ARTACHO† AND PHAN T. VUONG‡

Abstract. The boosted difference of convex functions algorithm (BDCA) was recently proposed for minimizing smooth difference of convex (DC) functions. BDCA accelerates the convergence of the classical difference of convex functions algorithm (DCA) thanks to an additional line search step. The purpose of this paper is twofold. First, we show that this scheme can be generalized and successfully applied to certain types of nonsmooth DC functions, namely, those that can be expressed as the difference of a smooth function and a possibly nonsmooth one. Second, we show that there is complete freedom in the choice of the trial step size for the line search, which is something that can further improve its performance. We prove that any limit point of the BDCA iterative sequence is a critical point of the problem under consideration and that the corresponding objective value is monotonically decreasing and convergent. The global convergence and convergence rate of the iterations are obtained under the Kurdyka–Łojasiewicz property. Applications and numerical experiments for two problems in data science are presented, demonstrating that BDCA outperforms DCA. Specifically, for the minimum sum-of-squares clustering problem, BDCA was on average 16 times faster than DCA, and for the multidimensional scaling problem, BDCA was 3 times faster than DCA.

Key words. difference of convex functions, boosted difference of convex functions algorithm, Kurdyka–Łojasiewicz property, clustering problem, multidimensional scaling problem

AMS subject classifications. 65K05, 65K10, 90C26, 47N10

DOI. 10.1137/18M123339X

1. Introduction.

In this paper, we are interested in the following difference of convex (DC) optimization problem

$$(\mathcal{P}) \quad \underset{x\in\mathbb{R}^m}{\text{minimize}}\; g(x) - h(x) =: \phi(x), \tag{1.1}$$

where $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ and $h : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are proper convex functions, with the conventions

$$(+\infty) - (+\infty) = +\infty,$$
$$(+\infty) - \lambda = +\infty \quad \text{and} \quad \lambda - (+\infty) = -\infty \quad \forall \lambda \in \,]-\infty, +\infty[.$$

For solving $(\mathcal{P})$, one usually applies the well-known DC algorithm (DCA) [21, 22, 34] (see section 3). DC programming and the DCA have been investigated and developed for more than 30 years [20]. The DCA has been successfully applied in many fields, such as machine learning, financial optimization, supply chain management, and telecommunication [19, 22, 20]. If both functions g and h are differentiable, then

*Received by the editors December 17, 2018; accepted for publication (in revised form) January 24, 2020; published electronically March 23, 2020.
https://doi.org/10.1137/18M123339X
Funding: The first author was supported by MICINN of Spain and ERDF of EU, as part of the Ramón y Cajal program (RYC-2013-13327), and the grants MTM2014-59179-C2-1-P and PGC2018-097960-B-C22. The second author was supported by the FWF (Austrian Science Fund), project M 2499-N32, and by the Vietnam National Foundation for Science and Technology Development (NAFOSTED), project 101.01-2019.320.
†Department of Mathematics, University of Alicante, Alicante, Spain (francisco.aragon@ua.es).
‡School of Mathematical Sciences, University of Southampton, University Road, SO17 1 BJ, Southampton, UK (t.v.phan@soton.ac.uk).

the boosted DC algorithm (BDCA) developed in [2] can be applied to accelerate the convergence of DCA. Numerical experiments with various biological data sets in [2] showed that BDCA outperforms DCA, being on average more than four times faster in both computational time and the number of iterations. This advantage has been also confirmed when applying BDCA to the indefinite kernel support vector machine problem [36].

The purpose of the present paper is to develop a version of BDCA when the function ϕ is not differentiable. Unfortunatelly, when g is not differentiable, the direction used by BDCA may no longer be a descent direction (see Example 3.4). For this reason, we shall restrict ourselves to the case where g is assumed to be differentiable but h is not. The motivation for this study comes from many applications of DC programming where the objective function is the difference of a smooth convex function and a nonsmooth convex function. We mention here the minimum sum-of-squares clustering problem [13], the bilevel hierarchical clustering problem [27], the multicast network design problem [15], and the multidimensional scaling (MDS) problem [18], among others.

The paper is organized as follows. In section 2, we recall some basic concepts and properties of convex analysis. As we are working with nonconvex and nonsmooth functions, we need some tools from variational analysis for generalized differentiability.

Our main contributions are in section 3, where we propose a nonsmooth version of the BDCA introduced in [2]. More precisely, we prove that the point generated by the DCA provides a descent direction for the objective function at this point, even at points where the function h is not differentiable. This is the key property allowing us to employ a simple line search along the descent direction, which permits us to achieve a larger decrease in the value of the objective function.

In section 4, we investigate the global convergence and convergence rate of the BDCA. The convergence analysis relies on the Kurdyka–Łojasiewicz inequality. These concepts of real algebraic geometry were introduced by Łojasiewicz [23] and Kurdyka [16] and later developed in the nonsmooth setting by Bolte et al. [9] and Attouch et al. [3], among many others [1, 4, 6, 8, 11, 28].

In section 5, we begin by introducing a self-adaptive strategy for choosing the trial step size for the line search step. We show that this strategy permits us to further improve the numerical results obtained in [2] for the above-mentioned problem arising in biochemistry, BDCA being almost 7 times faster than DCA on average. Next, we present an application of BDCA to two important classes of DC programming problems in engineering: the minimum sum-of-squares clustering problem and the MDS problem. We present some numerical experiments on large data sets, with both real and randomly generated data, which clearly show that BDCA outperforms DCA. Namely, on average, BDCA was 16 times faster than DCA for the minimum sum-of-squares clustering and 3 times faster for the MDS problems. We conclude the paper with some remarks and future research directions in the last section.

2. Preliminaries. Throughout this paper, the inner product of two vectors $x, y \in \mathbb{R}^m$ is denoted by $\langle x, y\rangle$, while $\|\cdot\|$ denotes the induced norm, defined by $\|x\| = \sqrt{\langle x, x\rangle}$. The closed ball of center x and radius $r > 0$ is denoted by $\mathbb{B}(x, r)$.

2.1. Tools of convex and variational analysis. In this subsection, we recall some basic concepts and results of convex analysis and generalized differentiation for nonsmooth functions, which will be used in what follows.

For an extended real-valued function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, the domain of f is the set

$$\operatorname{dom} f = \{x \in \mathbb{R}^m : f(x) < +\infty\}.$$

The function f is said to be proper if its domain is nonempty. It is said to be *convex* if

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \quad \forall x, y \in \mathbb{R}^m \text{ and } \lambda \in]0,1[,$$

and f is said to be *concave* if $-f$ is convex. Further, f is called *strongly convex* with modulus $\rho > 0$ if for all $x, y \in \mathbb{R}^m$ and $\lambda \in]0,1[$,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) - \frac{1}{2}\rho\lambda(1-\lambda)\|x-y\|^2,$$

or, equivalently, when $f - \frac{\rho}{2}\|\cdot\|^2$ is convex. The function f is said to be *coercive* if $f(x) \to +\infty$ whenever $\|x\| \to +\infty$. The gradient of a function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ which is differentiable at some point x in the interior of $\operatorname{dom} f$ is denoted by $\nabla f(x)$. We denote by $f'(x,d)$ the one-sided directional derivative of f at $x \in \operatorname{dom} f$ for the direction $d \in \mathbb{R}^m$, defined as

$$f'(x;d) := \lim_{t \downarrow 0} \frac{f(x+td) - f(x)}{t}.$$

A function $F : \mathbb{R}^m \to \mathbb{R}^m$ is said to be *monotone* when

$$\langle F(x) - F(y), x - y \rangle \geq 0 \quad \forall x, y \in \mathbb{R}^m.$$

Further, F is called *strongly monotone* with modulus $\rho > 0$ when

$$\langle F(x) - F(y), x - y \rangle \geq \rho\|x-y\|^2 \quad \forall x, y \in \mathbb{R}^m.$$

The function F is called *Lipschitz continuous* if there is some constant $L \geq 0$ such that

$$\|F(x) - F(y)\| \leq L\|x-y\| \quad \forall x, y \in \mathbb{R}^m,$$

and F is said to be locally Lipschitz continuous if, for every x in $\mathbb{R}^m$, there exists a neighborhood U of x such that F restricted to U is Lipschitz continuous.

We have the following well-known result (see, e.g., [33, Exercise 12.59]).

FACT 2.1. *A function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is strongly convex with modulus ρ if and only if ∂f is strongly monotone with modulus ρ.*

The *convex subdifferential* $\partial f(\bar{x})$ of a function f at $\bar{x} \in \mathbb{R}^m$ is defined at any point $\bar{x} \in \operatorname{dom} f$ by

$$\partial f(\bar{x}) = \{u \in \mathbb{R}^m \mid f(x) - f(\bar{x}) \geq \langle u, x - \bar{x} \rangle \, \forall x \in \mathbb{R}^m\}$$

and is empty otherwise.

When dealing with nonconvex and nonsmooth functions, we have to consider subdifferentials more general than the convex one. One of the most widely used constructions is the Clarke subdifferential, which can be defined in several (equivalent) ways (see, e.g., [12]). For a given locally Lipschitz continuous function $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, the *Clarke subdifferential* of f at $\bar{x}$ is given by

$$\partial_C f(\bar{x}) = \operatorname{co}\left\{\lim_{x \to \bar{x},\, x \notin \Omega_f} \nabla f(x)\right\},$$

where co stands for the convex hull and Ω_f denotes the set of Lebesgue measure zero (by Rademacher's theorem) where f fails to be differentiable. When f is also convex on a neighborhood of $\bar{x}$, then $\partial_C f(\bar{x}) = \partial f(\bar{x})$ (see [12, Proposition 2.2.7]).

Clarke subgradients are generalizations of the usual gradient of smooth functions. Indeed, if f is strictly differentiable at x, we have

$$\partial_C f(x) = \{\nabla f(x)\};$$

see [12, Proposition 2.2.4]. However, it should be noted that if f is only Fréchet differentiable at x, then $\partial_C f(x)$ can contain points other than $\nabla f(x)$ (see, e.g., [12, Example 2.2.3]).

The next basic formulas facilitate the calculation of the Clarke subdifferential.

FACT 2.2 (basic calculus). *The following assertions hold:*

(i) *For any scalar s, one has*

$$\partial_C(sf)(x) = s\partial_C f(x).$$

(ii) $\partial_C(f+g)(x) \subset \partial_C f(x) + \partial_C g(x)$, *and equality holds if either f or g is strictly differentiable.*

Proof. See [12, Propositions 2.3.1 and 2.3.3]. For the last assertion, see [12, Corollary 1, p. 39]. ∎

2.2. Assumptions. Throughout this paper, the following two assumptions are made.

Assumption 1. Both functions g and h are strongly convex with modulus $\rho > 0$.

Assumption 2. The function h is subdifferentiable at every point in $\operatorname{dom} h$, i.e., $\partial h(x) \neq \emptyset$ for all $x \in \operatorname{dom} h$. The function g is continuously differentiable on an open set containing $\operatorname{dom} h$ and

$$\inf_{x \in \mathbb{R}^m} \phi(x) > -\infty. \tag{2.1}$$

Under these assumptions, the next necessary optimality condition holds.

FACT 2.3 (first-order necessary optimality condition). *If $x^* \in \operatorname{dom} \phi$ is an optimal solution of problem $(\mathcal{P})$ in* (1.1)*, then*

$$\partial h(x^*) = \{\nabla g(x^*)\}. \tag{2.2}$$

Proof. See [35, Theorem 3′]. ∎

Any point satisfying condition (2.2) is called a *stationary point* of $(\mathcal{P})$. One says that $\bar{x}$ is a *critical point* of $(\mathcal{P})$ if

$$\nabla g(\bar{x}) \in \partial h(\bar{x}).$$

It is obvious that every stationary point x^* is a critical point, but the converse is not true in general.

Example 2.4. Consider the DC function $\phi : \mathbb{R}^m \to \mathbb{R}$ defined for $x \in \mathbb{R}^m$ by

$$\phi(x) := \|x\|^2 + \sum_{i=1}^m x_i - \sum_{i=1}^m |x_i|.$$

It is not difficult to check that ϕ has 2^m critical points, namely, any $x \in \{-1, 0\}^m$, and only one stationary point $x^* := (-1, -1, \ldots, -1)$, which is the global minimum of ϕ.

3. DCA and BDCA. The key idea of the DCA to solve problem $(\mathcal{P})$ in (1.1) is to approximate the concave part $-h$ of the objective function ϕ by its affine majorization, and then minimize the resulting convex function. The algorithm proceeds as follows.

DCA [22]

1. Let x_0 be any initial point and set $k := 0$.
2. Select $u_k \in \partial h(x_k)$ and solve the strongly convex optimization problem

$$(\mathcal{P}_k) \quad \underset{x\in\mathbb{R}^m}{\text{minimize}}\ g(x) - \langle u_k, x\rangle$$

   to obtain its unique solution y_k.
3. If $y_k = x_k$ then STOP and RETURN x_k, otherwise set $x_{k+1} := y_k$, set $k := k + 1$, and go to step 2.

Let us introduce the algorithm we propose for solving problem $(\mathcal{P})$, BDCA. The algorithm is a nonsmooth version of the one proposed in [2], except for a small but relevant modification in step 4, where now we give total freedom to the initial value for the backtracking line search used for finding an appropriate value of the step size λ_k. In section 5, we demonstrate that this seemingly minor change permits *smarter choices* of the initial value than simply using a constant value $\overline{\lambda}$. We have also replaced λ_k in the right-hand side of the line search inequality by λ_k^2, which allows us to remove the inconvenient assumption $\rho > \alpha$ (see [2, Remark 3] for more details).

BDCA

1. Fix $\alpha > 0$ and $0 < \beta < 1$. Let x_0 be any initial point and set $k := 0$.
2. Select $u_k \in \partial h(x_k)$ and solve the strongly convex optimization problem

$$(\mathcal{P}_k) \quad \underset{x\in\mathbb{R}^m}{\text{minimize}}\ g(x) - \langle u_k, x\rangle$$

   to obtain its unique solution y_k.
3. Set $d_k := y_k - x_k$. If $d_k = 0$, STOP and RETURN x_k. Otherwise, go to step 4.
4. Choose any $\overline{\lambda}_k \geq 0$. Set $\lambda_k := \overline{\lambda}_k$.
   WHILE $\phi(y_k + \lambda_k d_k) > \phi(y_k) - \alpha\lambda_k^2\|d_k\|^2$ DO $\lambda_k := \beta\lambda_k$.
5. Set $x_{k+1} := y_k + \lambda_k d_k$. If $x_{k+1} = x_k$ then STOP and RETURN x_k, otherwise set $k := k + 1$, and go to step 2.

Observe that if one sets $\overline{\lambda}_k = 0$, the iterations of the BDCA and the DCA coincide. Hence, our convergence results for the BDCA apply in particular to the DCA. In the following proposition we show that $d_k := y_k - x_k$ is a descent direction for ϕ at y_k. Since the value of ϕ is always reduced at y_k with respect to that at x_k, one can achieve a larger decrease by moving along the direction d_k. This simple fact, which is the key idea of the BDCA, improves the performance of the DCA in many applications (see section 5).

PROPOSITION 3.1. *For all $k \in \mathbb{N}$, the following holds:*
(i) $\phi(y_k) \leq \phi(x_k) - \rho\|d_k\|^2$;
(ii) $\phi'(y_k; d_k) \leq -\rho\|d_k\|^2$;

(iii) *there is some* $\delta_k > 0$ *such that*

$$\phi\left(y_k + \lambda d_k\right) \leq \phi(y_k) - \alpha\lambda^2\|d_k\|^2 \quad \forall \lambda \in [0, \delta_k],$$

so the backtracking step 4 *of BDCA terminates finitely.*

Proof. The proof of (i) is similar to the one of [2, Proposition 3] and is therefore omitted. To prove (ii), pick any $v \in \partial h(y_k)$. Note that the one-sided directional derivative $\phi'(y_k; d_k)$ is given by

$$\begin{aligned}\phi'(y_k; d_k) &= \lim_{t\downarrow 0} \frac{\phi(y_k + td_k) - \phi(y_k)}{t} \\ &= \lim_{t\downarrow 0} \frac{g(y_k + td_k) - g(y_k)}{t} - \lim_{t\downarrow 0} \frac{h(y_k + td_k) - h(y_k)}{t} \\ &\leq \langle \nabla g(y_k), d_k\rangle - \langle v, d_k\rangle \qquad (3.1)\end{aligned}$$

by convexity of h. Since y_k is the unique solution of the strongly convex problem $(\mathcal{P}_k)$, we have

$$\nabla g(y_k) = u_k \in \partial h(x_k).$$

The function h is strongly convex with constant ρ. This implies, by Fact 2.1, that ∂h is strongly monotone with constant ρ. Therefore, since $v \in \partial h(y_k)$, it holds that

$$\langle u_k - v, x_k - y_k\rangle \geq \rho\|x_k - y_k\|^2.$$

Hence

$$\langle \nabla g(y_k) - v, d_k\rangle = \langle u_k - v, y_k - x_k\rangle \leq -\rho\|d_k\|^2,$$

and the proof follows by combining the last inequality with (3.1).

Finally, to prove (iii), if $d_k = 0$ there is nothing to prove. Otherwise, we have

$$\lim_{\lambda\downarrow 0} \frac{\phi(y_k + \lambda d_k) - \phi(y_k)}{\lambda} = \phi'(y_k; d_k) \leq -\rho\|d_k\|^2 < -\frac{\rho}{2}\|d_k\|^2 < 0.$$

Hence, there is some $\widetilde{\lambda}_k > 0$ such that

$$\frac{\phi(y_k + \lambda d_k) - \phi(y_k)}{\lambda} \leq -\frac{\rho}{2}\|d_k\|^2 \quad \forall \lambda \in \left]0, \widetilde{\lambda}_k\right],$$

that is,

$$\phi(y_k + \lambda d_k) \leq \phi(y_k) - \frac{\rho\lambda}{2}\|d_k\|^2 \quad \forall \lambda \in \left]0, \widetilde{\lambda}_k\right].$$

Setting $\delta_k := \min\left\{\widetilde{\lambda}_k, \frac{\rho}{2\alpha}\right\}$, we obtain

$$\phi(y_k + \lambda d_k) \leq \phi(y_k) - \alpha\lambda^2\|d_k\|^2 \quad \forall \lambda \in]0, \delta_k],$$

which completes the proof. ∎

Remark 3.2.

(i) When the function h is differentiable, BDCA uses the same direction as the Mine–Fukushima algorithm [24], since $y_k + \lambda d_k = x_k + (1 + \lambda)d_k$. The algorithm they propose is computationally undesirable in the sense that it uses an

exact line search. This was later fixed in the Fukushima–Mine algorithm [14] by considering an Armijo type rule for choosing the step size

$$x_{k+1} = x_k + \beta^l d_k = \beta^l y_k + \left(1 - \beta^l\right) x_k$$

for some $0 < \beta < 1$ and some nonnegative integer l. Since $0 < \beta < 1$, the step size $\lambda = \beta^l - 1$ chosen by the Fukushima–Mine algorithm [14] is always less than or equal to zero, while in BDCA, only step sizes $\lambda \in \left]0, \overline{\lambda}_k\right]$ are explored. Also, the Armijo rule differs, as BDCA searches for some λ_k such that $\phi(y_k + \lambda_k d_k) \leq \phi(y_k) - \alpha\lambda_k^2\|d_k\|^2$, while the Fukushima–Mine algorithm requires $\phi(x_k + \beta^l d_k) \leq \phi(x_k) - \alpha\beta^l\|d_k\|^2$.

(ii) We know from Proposition 3.1 that

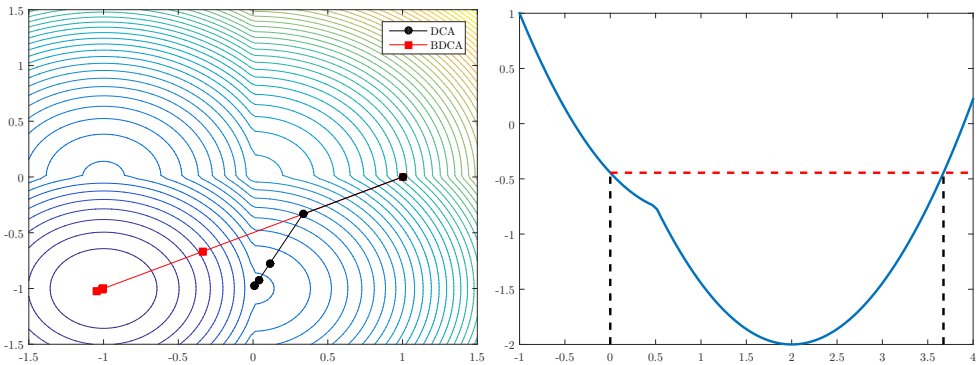
$$\phi\left(y_k + \lambda d_k\right) \leq \phi(y_k) - \alpha\lambda^2\|d_k\|^2 \leq \phi(x_k) - \left(\rho + \alpha\lambda^2\right)\|d_k\|^2;$$

thus, BDCA results in a larger decrease in the value of ϕ at each iteration than DCA. As a result, we can expect BDCA to converge faster than DCA.

Example 3.3 (Example 2.4 revisited). Consider again the function defined in Example 2.4 for $m = 2$. The function ϕ can be expressed as a DC function of type (1.1) with strongly convex terms by taking, for instance,

$$g(x,y) = \frac{3}{2}\left(x^2 + y^2\right) + x + y \quad \text{and} \quad h(x,y) = |x| + |y| + \frac{1}{2}\left(x^2 + y^2\right).$$

In Figure 1(a) we show the iterations generated by DCA and BDCA from the same starting point $(x_0, y_0) = (1, 0)$, with $\alpha = 0.1$, $\beta = 0.6$, and $\overline{\lambda}_k = 1$ for all k. Not only does BDCA obtain a larger decrease than DCA in the value of ϕ at each iteration, but also the line search helps the sequence generated escape from the stationary point $(0, -1)$, which is not even a local minimum. As the function h is not differentiable at (x_0, y_0), there is freedom in the choice of the point in $\partial h(x_0, y_0) = \{2\} \times [-1, 1]$ (we took the point $(2, 0)$). In Figure 1(b) we plot the value of the function in the line search procedure of BDCA at the first iteration. The value $\lambda = 0$ corresponds to the next iteration chosen by DCA, while BDCA choses $\lambda > 0$, which permits us to achieve an additional decrease in the value of ϕ.



(a) Iterations generated by DCA and BDCA (b) Line search of BDCA at the starting point

FIG. 1. *Illustration of Example* 3.3.

TABLE 1
For 1 million random starting points in $[-1.5, 1.5]^2$, we count the sequences generated by DCA, BDCA, the Oliveira–Tcheou algorithm (OTA) [29], and the Banert–Boţ algorithm (BBA) [6], converging to each of the four stationary points.

	$(-1,-1)$	$(-1,0)$	$(0,-1)$	$(0,0)$
DCA	249,763	249,841	250,204	250,192
BDCA	1,000,000	0	0	0
OTA	1,000,000	0	0	0
BBA	250,980	249,377	249,831	249,812

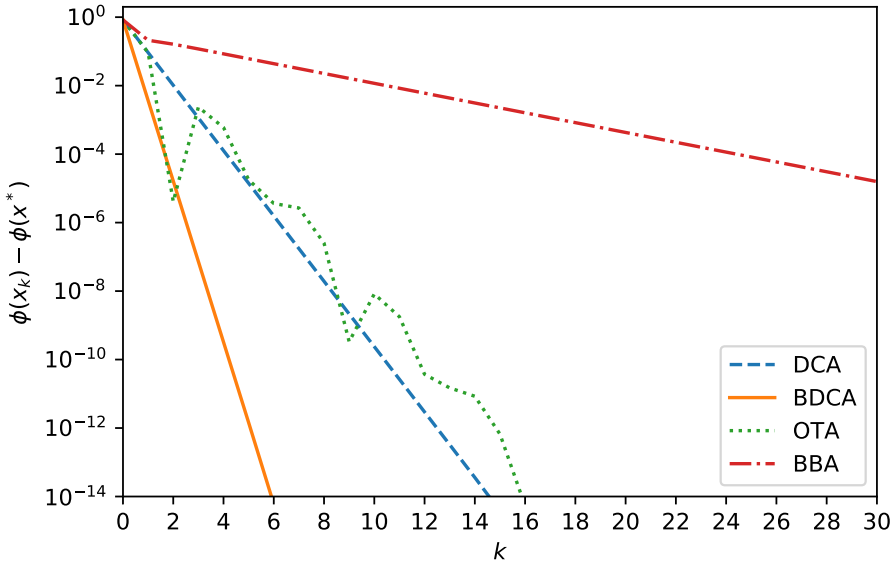


FIG. 2. *Comparison of the objective function value (using logarithmic scale) of DCA, BDCA, the Oliveira–Tcheou algorithm (OTA) [29], and the Banert–Boţ algorithm (BBA) [6] for a particular random instance where the four algorithms converge to the global minimum $x^* = (-1,-1)$.*

To demonstrate that, indeed, the line search procedure of BDCA helps the iterations escape from stationary points that are not critical points, we show in Table 1 the results of running both algorithms for 1 million random starting points. For only 25% of the starting points, DCA finds the optimal solution, while BDCA finds it in 100% of the instances.

In [29], Oliveira and Tcheou have recently introduced a modification of DCA by adding the inertial term $\gamma(x_k - x_{k-1})$ to u_k in the subproblem $(\mathcal{P}_k)$, where $\gamma \in [0, \frac{\rho}{2}[$. Although the sequence of objective values of the resulting algorithm is no longer monotone, as can be observed in Figure 2, the numerical experiments in [29] show that this term can be beneficial for escaping from stationary points that are not critical. This is also the case here: although the scheme is slower than BDCA, the algorithm finds the optimal solution in 100% of the instances.

Last, we test the performance of a double-proximal gradient algorithm which was recently introduced by Banert and Boţ in [6]. This algorithm generates both a primal and a dual sequence by using two proximal steps at each iteration. In our setting, where the function g is smooth and its gradient is Lipschitz continuous with constant $\frac{1}{\beta} = 3$, only one proximal step is needed, and the scheme reads as follows: Let $(x_0, z_0) \in \mathbb{R}^2 \times \mathbb{R}^2$ be a pair of starting points, and set for $k \geq 0$

$$x_{k+1} = x_k + \gamma_k z_k - \gamma_k \nabla g(x_k), \tag{3.2}$$

$$\begin{aligned} z_{k+1} &= z_k + \mu_k x_{k+1} - \mu_k \mathrm{prox}_{\mu_k^{-1} h}\left(\mu_k^{-1} z_k + x_{k+1}\right) \\ &= z_k + \mu_k x_{k+1} - \mu_k \operatorname*{argmin}_{u \in \mathbb{R}^2} \left\{ \frac{1}{\mu_k} h(u) + \frac{1}{2} \left\| \mu_k^{-1} z_k + x_{k+1} - u \right\|^2 \right\}, \end{aligned} \tag{3.3}$$

where μ_k and γ_k are some positive step sizes. To ensure that the hypotheses of the convergence results from [6] are satisfied, we take $\mu_k = \gamma_k = 0.3 < \beta$. We observe in Table 1 that this algorithm also often gets stuck in stationary points. Although the proximal step can be explicitly computed for this example, which is not always the case and can be time-consuming, the Banert–Boţ algorithm turns out to be the slowest among the three; see a particular random instance in Figure 2.

The next example complements the one given in [2, Remark 1]. It shows that the direction used by BDCA can be an ascent direction at y_k even when this point is not the global minimum of ϕ. Thus, Proposition 3.1 does not remain valid when g is not differentiable, and the scheme cannot be further extended.

Example 3.4 (failure of BDCA when g is not differentiable). Consider now the modification of the previous example

$$g(x,y) = -\frac{5}{2}x + x^2 + y^2 + |x| + |y| \quad \text{and} \quad h(x,y) = \frac{1}{2}\left(x^2 + y^2\right),$$

so that now h is differentiable but g is not. Let $(x_0, y_0) = (\frac{1}{2}, 1)$. Then, the next point generated by DCA is $(x_1, y_1) = (1, 0)$ and $d_0 := (x_1, y_1) - (x_0, y_0) = (\frac{1}{2}, -1)$ is not a descent direction for ϕ at (x_1, y_1). Indeed, one can easily check that

$$\phi'((x_1, y_1); d_0) = \lim_{t \downarrow 0} \frac{\phi\left((1,0) + t\left(\frac{1}{2}, -1\right)\right) - \phi(1,0)}{t} = \frac{3}{4};$$

see Figure 3. Actually, it holds that

$$\phi\left((x_1, y_1) + t d_0\right) - \phi(x_1, y_1) = \frac{5t^2}{8} + \frac{3t}{4},$$

so $\phi\left((x_1, y_1) + t d_0\right) > \phi(x_1, y_1)$ for all $t > 0$.

In contrast with the example in [2, Remark 1], observe that here (x_1, y_1) is not the global minimum of ϕ. In fact, the iterates generated by DCA converge to the global minimum of ϕ, as shown in Figure 3(a).

As proved next, the failure of BDCA shown in Example 3.4 can occur only for $n \geq 2$.

PROPOSITION 3.5. *Let $\phi = g - h$, where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$ are convex and h is differentiable. If $h'(x) \in \partial g(y)$ and $0 \notin \partial_C \phi(y)$, then $\phi'(y; y - x) < 0$.*

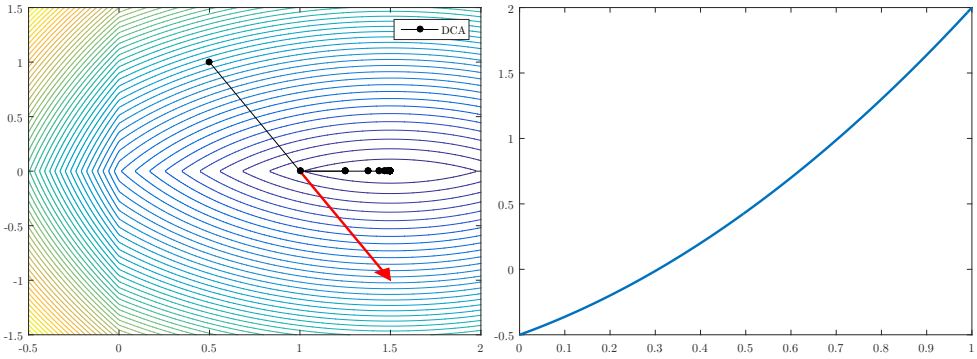
Proof. First, observe that

$$\phi'(y; y - x) = (y - x) \sup_{z \in \partial g(y)} \left\{ z - h'(y) \right\}.$$

Since h is convex, one has

$$\left(h'(x) - h'(y)\right)(x - y) \geq 0.$$

Suppose that $x - y > 0$. Then, $h'(x) \geq h'(y)$. Since $h'(y) \notin \partial g(y)$ and $\partial g(y)$ is convex, we deduce that $h'(y) < z$ for all $z \in \partial g(y)$, which implies $\phi'(y; y - x) < 0$. A similar argument shows that $\phi'(y; y - x) < 0$ when $x - y < 0$. This concludes the proof. ∎



(a) Iterations generated by DCA and search direction of BDCA at $(1, 0)$ (b) Line search of BDCA at the point (1,0)

FIG. 3. *Illustration of Example* 3.4.

We are now in a position to state our first convergence result of the iterative sequence generated by BDCA, whose statement coincides with [2, Proposition 5]. The first part of its proof requires some small adjustments due to the nonsmoothness of h.

THEOREM 3.6. *For any $x_0 \in \mathbb{R}^m$, either BDCA returns a critical point of $(\mathcal{P})$ or it generates an infinite sequence such that the following holds:*

(i) *$\phi(x_k)$ is monotonically decreasing and convergent to some ϕ^*.*

(ii) *Any limit point of $\{x_k\}$ is a critical point of $(\mathcal{P})$. If in addition ϕ is coercive, then there exists a subsequence of $\{x_k\}$ which converges to a critical point of $(\mathcal{P})$.*

(iii) *$\sum_{k=0}^{+\infty} \|d_k\|^2 < +\infty$. Further, if there is some $\overline{\lambda}$ such that $\lambda_k \leq \overline{\lambda}$ for all k, then $\sum_{k=0}^{+\infty} \|x_{k+1} - x_k\|^2 < +\infty$.*

Proof. If BDCA stops at step 3 and returns x_k, then $x_k = y_k$. Because y_k is the unique solution of the strongly convex problem $(\mathcal{P}_k)$, we have

$$\nabla g(x_k) = u_k \in \partial h(x_k),$$

i.e., x_k is a critical point of $(\mathcal{P})$. Otherwise, by Proposition 3.1 and step 4 of BDCA, we have

$$\phi(x_{k+1}) \leq \phi(y_k) - \alpha\lambda_k^2\|d_k\|^2 \leq \phi(x_k) - \left(\alpha\lambda_k^2 + \rho\right)\|d_k\|^2. \tag{3.4}$$

Therefore, the sequence $\{\phi(x_k)\}$ converges to some ϕ^*, since is monotonically decreasing and bounded from below by (2.1). This proves (i). As a consequence, we obtain

$$\phi(x_{k+1}) - \phi(x_k) \to 0,$$

which implies $\|d_k\|^2 = \|y_k - x_k\|^2 \to 0$, by (3.4).

If $\bar{x}$ is a limit point of $\{x_k\}$, there exists a subsequence $\{x_{k_i}\}$ converging to $\bar{x}$. Then, as $\|y_{k_i} - x_{k_i}\| \to 0$, we have $y_{k_i} \to \bar{x}$. Since ∇g is continuous, we get

$$u_{k_i} = \nabla g(y_{k_i}) \to \nabla g(\bar{x}).$$

Hence, we deduce $\nabla g(\bar{x}) \in \partial h(\bar{x})$, thanks to the closedness of the graph of ∂h (see [32, Theorem 24.4]). When ϕ is coercive, by (i), the sequence $\{x_k\}$ must be bounded, which implies the rest of the claim in (ii).

The proof of (iii) is similar to that of [2, Proposition 5(iii)] and is thus omitted. ∎

Remark 3.7. In our approach, both functions g and h are assumed to be strongly convex with constant $\rho > 0$. It is well known that the performance of DCA heavily depends on the decomposition of the objective function [22, 31]. There is an infinite number of ways of doing this and it is challenging to find a "good" one [31]. To get rid of this assumption, one could add a proximal term $\frac{\rho_k}{2}\|x - x_k\|^2$ to the objective of the convex optimization subproblem $(\mathcal{P}_k)$ in step 2, as done in the proximal point algorithm (see [14]). This technique is employed in the proximal DCA; see [1, 6, 20, 26]. With some minor adjustments in the proofs, it is easy to show that the resulting algorithm satisfies both Proposition 3.1 and Theorem 3.6.

4. Convergence under the Kurdyka–Łojasiewicz property. In this section, we prove the convergence of the sequence generated by BDCA as long as the sequence has a cluster point at which ϕ satisfies the strong Kurdyka–Łojasiewicz inequality [23, 16, 9] and ∇g is locally Lipschitz. As we shall see, under some additional assumptions, linear convergence can also be guaranteed.

DEFINITION 4.1. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a locally Lipschitz function. We say that f satisfies the* strong Kurdyka–Łojasiewicz inequality *at $x^* \in \mathbb{R}^m$ if there exist $\eta \in]0, +\infty[$, a neighborhood U of x^*, and a concave function $\varphi : [0, \eta] \to [0, +\infty[$ such that*

(i) $\varphi(0) = 0$;
(ii) φ *is of class* $\mathcal{C}^1$ *on* $]0, \eta[$;
(iii) $\varphi' > 0$ *on* $]0, \eta[$;
(iv) *for all* $x \in U$ *with* $f(x^*) < f(x) < f(x^*) + \eta$ *we have*

$$\varphi'(f(x) - f(x^*)) \operatorname{dist}(0, \partial_C f(x)) \geq 1. \tag{4.1}$$

For strictly differentiable functions the latter reduces to the standard definition of the Kurdyka–Łojasiewicz inequality. Bolte et al. [9, Theorem 14] show that *definable functions* satisfy the strong Kurdyka–Łojasiewicz inequality at each point in $\operatorname{dom} \partial_C f$, which covers a large variety of practical cases.

Remark 4.2. Although the concavity of the function φ does not explicitly appear in the statement of [9, Theorem 14], the function φ can be chosen to be concave (since φ is o-minimal by construction, its second derivative exists and maintains the sign on an interval $]0, \delta[$, and this sign is necessarily negative). If the function f is not o-minimal but is convex and satisfies the Kurdyka–Łojasiewicz inequality with a function φ which is not concave, then f also satisfies the Kurdyka–Łojasiewicz inequality with another function Ψ which is concave (see [10, Theorem 29]).

THEOREM 4.3. *For any $x_0 \in \mathbb{R}^m$, consider the sequence $\{x_k\}$ generated by the BDCA. Suppose that $\{x_k\}$ has a cluster point x^*, that ∇g is locally Lipschitz continuous around x^*, and that ϕ satisfies the strong Kurdyka–Łojasiewicz inequality at x^*. Then $\{x_k\}$ converges to x^*, which is a critical point of $(\mathcal{P})$.*

Proof. By Theorem 3.6, we have $\lim_{k\to+\infty} \phi(x_k) = \phi^*$. Let x^* be a cluster point of the sequence $\{x_k\}$. Then, there exists a subsequence $\{x_{k_i}\}$ of $\{x_k\}$ such that $\lim_{i\to+\infty} x_{k_i} = x^*$. Thanks to the continuity of ϕ, we deduce

$$\phi(x^*) = \lim_{i\to+\infty} \phi(x_{k_i}) = \lim_{k\to\infty} \phi(x_k) = \phi^*.$$

Hence, the function ϕ is finite and has the same value ϕ^* at every cluster point of $\{x_k\}$.

If $\phi(x_k) = \phi^*$ for some $k > 1$, then $\phi(x_k) = \phi(x_{k+1})$, because the sequence $\{\phi(x_k)\}$ is decreasing. From (3.4), we deduce that $d_k = 0$, so BDCA terminates after a finite number of steps. Thus, from now on, we assume that $\phi(x_k) > \phi^*$ for all k.

Since ∇g is locally Lipschitz around x^*, there exist some constants $L \geq 0$ and $\delta_1 > 0$ such that

$$\|\nabla g(x) - \nabla g(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{B}(x^*, \delta_1). \tag{4.2}$$

Further, since ϕ satisfies the strong Kurdyka–Łojasiewicz inequality at x^*, there exist $\eta \in]0, +\infty[$, a neighborhood U of x^*, and a continuous and concave function $\varphi : [0, \eta] \to [0, +\infty[$ such that for every $x \in U$ with $\phi(x^*) < \phi(x) < \phi(x^*) + \eta$, we have

$$\varphi'(\phi(x) - \phi(x^*)) \operatorname{dist}(0, \partial_C \phi(x)) \geq 1. \tag{4.3}$$

Take δ_2 small enough that $\mathbb{B}(x^*, \delta_2) \subset U$ and set $\delta := \frac{1}{2}\min\{\delta_1, \delta_2\}$. Let

$$K := \max_{\lambda \geq 0} \frac{L(1+\lambda)}{\alpha\lambda^2 + \rho}, \tag{4.4}$$

which is attained at $\hat{\lambda} = -1 + \sqrt{1 + \rho/\alpha}$. Since $\lim_{i\to+\infty} x_{k_i} = x^*$, $\lim_{i\to+\infty} \phi(x_{k_i}) = \phi^*$, $\phi(x_k) > \phi^*$ for all k, and φ is continuous, we can find an index N large enough such that

$$x_N \in \mathbb{B}(x^*, \delta), \quad \phi^* < \phi(x_N) < \phi^* + \eta \tag{4.5}$$

and

$$\|x_N - x^*\| + K\varphi(\phi(x_N) - \phi^*) < \delta. \tag{4.6}$$

By Theorem 3.6(iii), we know that $d_k = y_k - x_k \to 0$. Then, taking a larger N if needed, we can ensure that

$$\|y_k - x_k\| \leq \delta \quad \forall k \geq N.$$

For all $k \geq N$ such that $x_k \in \mathbb{B}(x^*, \delta)$, we have

$$\|y_k - x^*\| \leq \|y_k - x_k\| + \|x_k - x^*\| \leq 2\delta \leq \delta_1;$$

then, using (4.2), we obtain

$$\|\nabla g(y_k) - \nabla g(x_k)\| \leq L\|y_k - x_k\| = \frac{L}{1+\lambda_k}\|x_{k+1} - x_k\|.$$

On the other hand, we have from the optimality condition of $(\mathcal{P}_k)$ that

$$\nabla g(y_k) = u_k \in \partial h(x_k),$$

which implies, by Fact 2.2,

$$\nabla g(y_k) - \nabla g(x_k) \in \partial h(x_k) - \nabla g(x_k) = \partial_C(-\phi(x_k)) = -\partial_C \phi(x_k). \tag{4.7}$$

Therefore,

$$\operatorname{dist}(0, \partial_C \phi(x_k)) \leq \|\nabla g(y_k) - \nabla g(x_k)\| \leq \frac{L}{1+\lambda_k}\|x_{k+1} - x_k\|. \tag{4.8}$$

For all $k \geq N$ such that $x_k \in \mathbb{B}(x^*, \delta)$ and $\phi^* < \phi(x_k) < \phi^* + \eta$, it follows from (4.8), the concavity of φ, (4.3), and (3.4) that

$$\begin{aligned}
\frac{L}{1+\lambda_k}\|x_k - x_{k+1}\| &\left(\varphi\left(\phi(x_k) - \phi^*\right) - \varphi\left(\phi(x_{k+1}) - \phi^*\right)\right) \\
&\geq \operatorname{dist}(0, \partial_C \phi(x_k))\left(\varphi\left(\phi(x_k) - \phi^*\right) - \varphi\left(\phi(x_{k+1}) - \phi^*\right)\right) \\
&\geq \operatorname{dist}(0, \partial_C \phi(x_k))\, \varphi'\left(\phi(x_k) - \phi^*\right)\left(\phi(x_k) - \phi(x_{k+1})\right) \\
&\geq \phi(x_k) - \phi(x_{k+1}) \\
&\geq \left(\alpha\lambda_k^2 + \rho\right)\|y_k - x_k\|^2 = \frac{\alpha\lambda_k^2 + \rho}{(1+\lambda_k)^2}\|x_k - x_{k+1}\|^2,
\end{aligned}$$

which implies, by (4.4), that

$$\begin{aligned}
\|x_k - x_{k+1}\| &\leq \frac{L(1+\lambda_k)}{\alpha\lambda_k^2 + \rho}\left(\varphi\left(\phi(x_k) - \phi^*\right) - \varphi\left(\phi(x_{k+1}) - \phi^*\right)\right) \\
&\leq K\left(\varphi\left(\phi(x_k) - \phi^*\right) - \varphi\left(\phi(x_{k+1}) - \phi^*\right)\right). \qquad (4.9)
\end{aligned}$$

We prove by induction that $x_k \in \mathbb{B}(x^*, \delta)$ for all $k \geq N$. Indeed, from (4.5) the claim holds for $k = N$. We suppose that it also holds for $k = N, N+1, \ldots, N+p-1$, with $p \geq 1$. Since $\{\phi(x_k)\}$ is a decreasing sequence converging to ϕ^*, our choice of N implies that $\phi^* < \phi(x_k) < \phi^* + \eta$ for all $k \geq N$. Then (4.9) is valid for $k = N, N+1, \ldots, N+p-1$. Hence,

$$\begin{aligned}
\|x_{N+p} - x^*\| &\leq \|x_N - x^*\| + \sum_{i=1}^{p}\|x_{N+i} - x_{N+i-1}\| \\
&\leq \|x_N - x^*\| + K\sum_{i=1}^{p}\left[\varphi\left(\phi(x_{N+i-1}) - \phi^*\right) - \varphi\left(\phi(x_{N+i}) - \phi^*\right)\right] \\
&\leq \|x_N - x^*\| + K\varphi\left(\phi(x_N) - \phi^*\right) < \delta,
\end{aligned}$$

where the last inequality follows from (4.6).

Thus, adding (4.9) from $k = N$ to P, we get

$$\sum_{k=N}^{P}\|x_{k+1} - x_k\| \leq K\varphi\left(\phi(x_N) - \phi^*\right),$$

and taking the limit as $P \to +\infty$, we conclude that

$$\sum_{k=1}^{+\infty}\|x_{k+1} - x_k\| < +\infty. \tag{4.10}$$

Therefore, $\{x_k\}$ is a Cauchy sequence, and since x^* is a cluster point of $\{x_k\}$, the whole sequence converges to x^*. By Theorem 3.6, x^* must be a critical point of $(\mathcal{P})$. ∎

Remark 4.4. In the proof of Theorem 4.3, we assume the *strong Kurdyka–Łojasiewicz inequality* in order to have Fact 2.2(i), which allows us to deduce the last equality in (4.7). Therefore, Theorem 4.3 remains valid if, instead of requiring that ϕ satisfies the strong Kurdyka–Łojasiewicz inequality, we assume that $-\phi$ satisfies the *Kurdyka–Łojasiewicz inequality* (which is defined in the same way but replacing in (4.1) the Clarke subdifferential by the limiting subdifferential). Another possibility would be to assume that ϕ satisfies the *symmetric Kurdyka–Łojasiewicz inequality*, which can be defined as in Definition 4.1 with (4.1) replaced by

$$\varphi'(f(x) - f(x^*)) \operatorname{dist}(0, \partial_0 f(x)) \geq 1, \tag{4.11}$$

where $\partial_0 f(x)$ is the *symmetric subdifferential* of f at x (see, e.g., [25, p. 171]), defined by

$$\partial_0 f(x) := \partial_L f(x) \cup [-\partial_L (-f(x))], \tag{4.12}$$

with $\partial_L f(x)$ denoting the *limiting subdifferential* of f at x. It is clear that $\partial_L f(x) \subset \partial_0 f(x) \subset \partial_C f(x)$ for locally Lipschitz functions. Moreover $\partial_0(-f(x)) = -\partial_0 f(x)$ (see [25, Exercise 1.75]), which is exactly what we need to deduce the last equality in (4.7). For more information on this, we recommend the discussion in [25, pp. 61–62].

Remark 4.5. Our convergence analysis in Theorem 4.3 shares a similar flavor with the general framework developed by Attouch, Bolte, and Svaiter in [5]. Indeed, in that paper, the authors provided a convergence analysis for any generic algorithm for solving nonsmooth nonconvex optimization problems satisfying two key properties:

(i) for each $k \in \mathbb{N}$: $\phi(x_{k+1}) + a\|x_{k+1} - x_k\|^2 \leq \phi(x_k)$;
(ii) for each $k \in \mathbb{N}$ there exists $w_{k+1} \in \partial_L \phi(x_{k+1})$ such that

$$\|w_{k+1}\| \leq b\|x_{k+1} - x_k\|, \tag{4.13}$$

where a and b are some positive constants. In our convergence analysis, while we also have the descent property (i) for BDCA (see (3.4)), it is not clear if (ii) holds. Instead of (ii), from (4.8) we can deduce a similar property: for each k large enough there exists $w_k \in \partial_C \phi(x_k)$ such that

$$\|w_k\| \leq b\|x_{k+1} - x_k\|. \tag{4.14}$$

This also opens the question of whether the general framework from Attouch, Bolte, and Svaiter [5] still holds when we replace (4.13) by (4.14). The answer seems to be positive, at least for DC programming.

Remark 4.6. In [28], Noll proposed an alternative algorithm for nonsmooth nonconvex optimization problems without requiring (4.13) or (4.14), where the convergence analysis also heavily depends on the strong Kurdyka–Łojasiewicz inequality. In addition, he provided an example where both (4.13) and (4.14) could fail. Another algorithm for DC programming whose convergence was analyzed under the Kurdyka–Łojasiewicz inequality is the double-proximal gradient algorithm by Banert and Boţ [6], which was briefly discussed in Example 3.3. This algorithm can be applied to more general DC problems of the type

$$\min_{x \in \mathbb{R}^m} \{g(x) + \phi(x) - h(Kx)\},$$

where g and h are proper, convex, and lower semicontinuous functions, ϕ is convex and differentiable with $\frac{1}{\beta}$-Lipschitz continuous gradient, for some $\beta > 0$, and K is

a linear mapping. Although neither strong convexity of g and h, nor smoothness of $g+\phi$ is required, this comes at the cost of needing to compute an additional proximal step. Further, when $g+\phi$ is smooth, the global Lipschitz continuity of the gradient of ϕ with a known constant is needed to guarantee the convergence of the algorithm, because the step size is bounded above by that constant (see [6, Theorem 1]). In contrast, only local Lipschitz continuity of ∇g is assumed in Theorem 4.3.

Remark 4.7. As mentioned before, if one sets $\overline{\lambda}_k = 0$ for all k, then BDCA becomes DCA. In this case, Theorem 4.3 is akin to [17, Theorem 3.4], where the function ϕ is assumed to be subanalytic. We also note that in this setting only one of the functions g or h needs to be strongly convex, since one can easily check that [2, Proposition 3] still holds, and Proposition 3.1(ii) is not needed anymore.

Next, we establish the convergence rate on the iterative sequence $\{x_k\}$ when ϕ satisfies the strong Kurdyka–Łojasiewicz inequality with $\varphi(t) = Mt^{1-\theta}$ for some $M > 0$ and $0 \leq \theta < 1$. Observe that this property holds for all globally subanalytic functions [9, Corollary 16], which covers many classes of functions in applications. We will employ the following useful lemma, whose proof appears within that of [4, Theorem 2] for specific values of α and β.

LEMMA 4.8 (see [2, Lemma 1]). *Let $\{s_k\}$ be a nonnegative sequence in $\mathbb{R}$ and let α, β be some positive constants. Suppose that $s_k \to 0$ and that the sequence satisfies*

$$s_k^\alpha \leq \beta(s_k - s_{k+1}) \quad \forall\ k \text{ sufficiently large.}$$

Then,

(i) *if $\alpha = 0$, the sequence $\{s_k\}$ converges to 0 in a finite number of steps;*
(ii) *if $\alpha \in]0, 1]$, the sequence $\{s_k\}$ converges linearly to 0 with rate $1 - \frac{1}{\beta}$;*
(iii) *if $\alpha > 1$, there exists $\eta > 0$ such that*

$$s_k \leq \eta k^{-\frac{1}{\alpha-1}} \quad \forall\ k \text{ sufficiently large.}$$

THEOREM 4.9. *Suppose that the sequence $\{x_k\}$ generated by the BDCA has the limit point x^*. Assume that ∇g is locally Lipschitz continuous around x^* and ϕ satisfies the strong Kurdyka–Łojasiewicz inequality at x^* with $\varphi(t) = Mt^{1-\theta}$ for some $M > 0$ and $0 \leq \theta < 1$. Then, the following convergence rates are guaranteed:*

(i) *if $\theta = 0$, then the sequence $\{x_k\}$ converges in a finite number of steps to x^*;*
(ii) *if $\theta \in \left]0, \frac{1}{2}\right]$, then the sequence $\{x_k\}$ converges linearly to x^*;*
(iii) *if $\theta \in \left]\frac{1}{2}, 1\right[$, then there exists a positive constant η such that*

$$\|x_k - x^*\| \leq \eta k^{-\frac{1-\theta}{2\theta-1}}$$

for all large k.

Proof. By (4.10), we know that $s_i := \sum_{k=i}^{+\infty} \|x_{k+1} - x_k\|$ is finite. Since $\|x_i - x^*\| \leq s_i$ by the triangle inequality, the rate of convergence of x_i to x^* can be deduced from the convergence rate of s_i to 0.

Adding (4.9) from i to P with $N \leq i \leq P$, we have

$$\sum_{k=i}^{P} \|x_{k+1} - x_k\| \leq K\varphi\left(\phi(x_i) - \phi^*\right) = KM(\phi(x_i) - \phi^*)^{1-\theta},$$

which implies that

$$s_i = \lim_{P \to +\infty} \sum_{k=i}^{P} \|x_{k+1} - x_k\| \leq KM(\phi(x_i) - \phi^*)^{1-\theta}. \tag{4.15}$$

Since ϕ satisfies the strong Kurdyka–Łojasiewicz inequality at x^* with $\varphi(t) = Mt^{1-\theta}$, we have

$$M(1-\theta)\left(\phi(x_i)-\phi^*\right)^{-\theta}\operatorname{dist}\left(0,\partial_C\phi(x_i)\right)\geq 1.$$

This and (4.8) imply

$$\begin{aligned}\left(\phi(x_i)-\phi^*\right)^{\theta} &\leq M(1-\theta)\operatorname{dist}\left(0,\partial_C\phi(x_i)\right)\\ &\leq \frac{ML(1-\theta)}{1+\lambda_i}\|x_{i+1}-x_i\|\\ &\leq ML(1-\theta)\|x_{i+1}-x_i\|. \qquad (4.16)\end{aligned}$$

Combining (4.15) and (4.16), we obtain

$$s_i^{\frac{\theta}{1-\theta}} \leq (KM)^{\frac{\theta}{1-\theta}}\left(\phi(x_i)-\phi^*\right)^{\theta} \leq ML(1-\theta)\left(KM\right)^{\frac{\theta}{1-\theta}}(s_i-s_{i+1}).$$

Applying Lemma 4.8, with $\alpha := \frac{\theta}{1-\theta}$ and $\beta := ML(1-\theta)\left(KM\right)^{\frac{\theta}{1-\theta}}$, we deduce the convergence rates in (i)–(iii). ∎

5. Applications and numerical experiments. The purpose of this section is to numerically compare the performance of DCA and BDCA. The lack of an explicit form of the proximal step of the Banert–Boţ algorithm (3.2)–(3.3), together with the relatively large size of our test problems, precludes the inclusion of this algorithm in our numerical experiments. We also tested the inertial DC algorithm of Oliveira and Tcheou [29] mentioned in Example 3.3, but we do not include any of the results because we did not observe any difference with respect to DCA. This is due to the small value of the inertial parameter, as it has to be set smaller than $\frac{\rho}{2}$ to guarantee the convergence of the algorithm, which is small in our test problems. All our codes were written in Python 2.7 and the tests were run on an Intel Core i7-4770 CPU 3.40GHz with 32GB RAM, under Windows 10 (64-bit).

In all the experiments in this section we use the following strategy for choosing the trial step size in step 4 of BDCA, which makes use of the previous step sizes. We emphasize that the convergence results in the previous sections apply to any possible choice of the trial step sizes $\overline{\lambda}_k$. This is in contrast with [2], where $\overline{\lambda}_k$ had to be chosen constantly equal to some fixed parameter $\overline{\lambda} > 0$.

Self-adaptive trial step size

Fix $\gamma > 1$. Set $\overline{\lambda}_0 = 0$. Choose some $\overline{\lambda}_1 > 0$ and obtain λ_1 by step 4 of BDCA.
For any $k \geq 2$:

1. IF $\lambda_{k-2} = \overline{\lambda}_{k-2}$ AND $\lambda_{k-1} = \overline{\lambda}_{k-1}$ THEN set $\overline{\lambda}_k := \gamma\lambda_{k-1}$; ELSE set $\overline{\lambda}_k := \lambda_{k-1}$.
2. Obtain λ_k from $\overline{\lambda}_k$ by step 4 of BDCA.

The latter *self-adaptive strategy* uses the step size that was chosen in the previous iteration as a new trial step size for the next iteration, except in the case where two consecutive trial step sizes were successful. In that case, the trial step size is increased by multiplying the previously accepted step size by $\gamma > 1$. Thus, we used a somewhat conservative strategy in our experiments, where two successful iterations are needed before increasing the trial step size. Other strategies could be easily considered. Since we set $\overline{\lambda}_0 = 0$, the first iteration is computed with DCA. In all our experiments we took $\gamma := 2$.

The self-adaptive strategy for the trial step size has two key advantages with respect to the *constant strategy* $\overline{\lambda}_k = \overline{\lambda} > 0$, which was used in [2]. The most important one is that we observed in our numerical tests almost a two times speed up in the running time of BDCA. The second advantage is that it is more adaptive and less sensitive to a wrong choice of the parameters. Indeed, in the constant strategy, a very large value of $\overline{\lambda}$ could make BDCA slow, due to the internal iterations needed in the backtracking step. On the other hand, a small value of $\overline{\lambda}$ would provide a trial step size that will be readily accepted but will result in a small advantage of BDCA against DCA.

In the next two subsections, we compare the performance of DCA and BDCA in two important nonsmooth problems in data analysis: the minimum sum-of-squares clustering problem and the MDS problem. Before doing that, let us begin by numerically demonstrating that the self-adaptive strategy permits us to further improve the results of BDCA in the smooth problem arising from the study of systems of biochemical reactions tested in [2], where BDCA was shown to be more than four times faster than DCA. To this aim, we used the same setting as in [2, section 5]. For each of five randomly selected starting points, we obtained the 1000th iterate of BDCA with constant trial step size strategy $\overline{\lambda} = 50$. Next, both BDCA with self-adaptive strategy (with $\beta = 0.1$) and DCA were run from the same starting point until they reached the same objective value as the one obtained by BDCA with constant strategy. Instead of presenting a table with the results, we show in Figure 4 the ratios of the running times between the three algorithms, which permits us to readily compare the three algorithms. On average, BDCA with self-adaptive strategy was 6.7 times faster than DCA and was 1.7 times faster than BDCA with constant strategy, which in turn was 4.2 times faster than DCA.

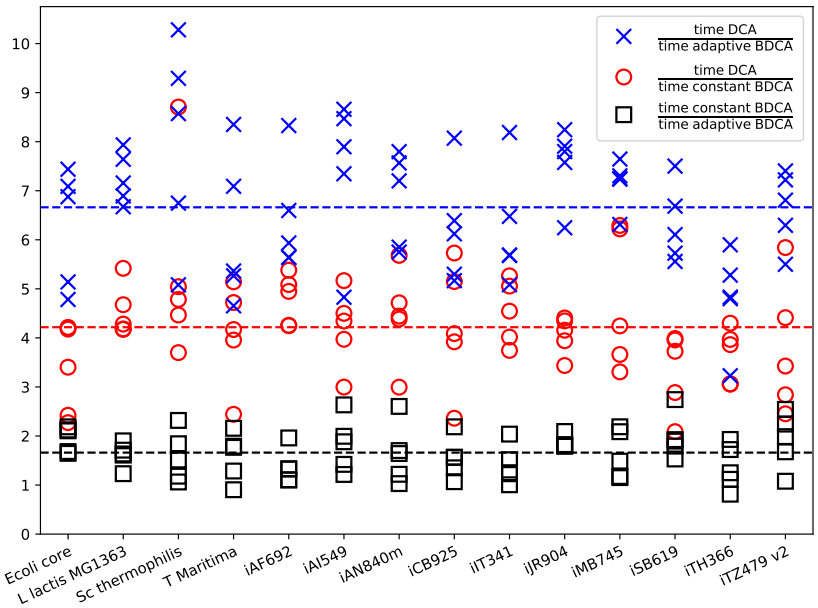


FIG. 4. *Ratios of the running times of DCA, BDCA with constant trial step size and BDCA with self-adaptive trial step size for finding a steady state of various biochemical reaction network models* [2]. *For each of the models, the algorithms were run using the same five random starting points. The average is represented with a dashed line.*

In the next two subsections we present various experiments with problems in data analysis. We consider two types of data: real and random. As real data, we use the geographic coordinates of the Spanish cities with more than 500 habitants.[1] The advantage of this relatively large data in $\mathbb{R}^2$ is that it permits us to visually illustrate some of the experiments.

5.1. The minimum sum-of-squares clustering problem. Clustering is an unsupervised technique for data analysis whose objective is to group a collection of objects into clusters based on similarity. This is among the most popular techniques in data mining and can be mathematically described as follows. Let $A = \{a^1, \dots, a^n\}$ be a finite set of points in $\mathbb{R}^m$, which represent the data points to be grouped. The goal is to partition A into k disjoint subsets $A^1, \dots, A^k$, called clusters, such that a clustering criterion is optimized.

There are many different criteria for the clustering problem. One of the most used is the *minimum sum-of-squares clustering* criterion, where one tries to minimize the Euclidean distance of each data point to the centroid of its cluster [7, 13, 30]. Thus, each cluster A_j is identified by its center (or centroid) $x^j \in \mathbb{R}^m, j = 1, \dots, k$. Letting $X := \left(x^1, \dots, x^k\right) \in \mathbb{R}^{m\times k}$, this gives rise to the following optimization problem:

$$\text{minimize} \quad \varphi(X, \omega) := \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \omega_{ij} \|x^j - a^i\|^2,$$

where the binary variables ω_{ij} express the assignment of the point a^i to the cluster j, i.e., $\omega_{ij} = 1$ if $a^i \in A^j$, and $\omega_{ij} = 0$ otherwise. This problem can be equivalently reformulated as the following nonsmooth nonconvex unconstrained optimization problem (see [13, 30]):

$$\underset{X\in\mathbb{R}^{m\times k}}{\text{minimize}}\ \phi(X) := \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,\dots,k} \|x^j - a^i\|^2. \tag{5.1}$$

As explained in [13, 30], we can write this problem as a DC problem of type (1.1) by taking

$$g(X) := \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \left\|x^j - a^i\right\|^2 + \frac{\rho}{2}\|X\|^2,$$

$$h(X) := \frac{1}{n} \sum_{i=1}^{n} \max_{j=1,\dots,k} \sum_{t=1,t\neq j}^{k} \left\|x^t - a^i\right\|^2 + \frac{\rho}{2}\|X\|^2,$$

for some $\rho \geq 0$, where $\|X\|$ is the Frobenius norm of X. Observe that both functions g and h are convex, and strongly convex if $\rho > 0$. Moreover, g is differentiable, and the subdifferential of h can be explicitly computed (see [30, p. 346] or [13, equation (3.21)]).

Experiment 5.1 (clustering the Spanish cities in the peninsula). *Consider the problem of finding a partition into five clusters of the* 4001 *Spanish cities in the peninsula with more than* 500 *residents. For illustrating the difference between the iterations of DCA and BDCA, we present in Figure* 5 *the result of applying* 7 *iterations of DCA*

[1] The data can be retrieved from the Spanish National Center of Geographic Information at http://centrodedescargas.cnig.es.

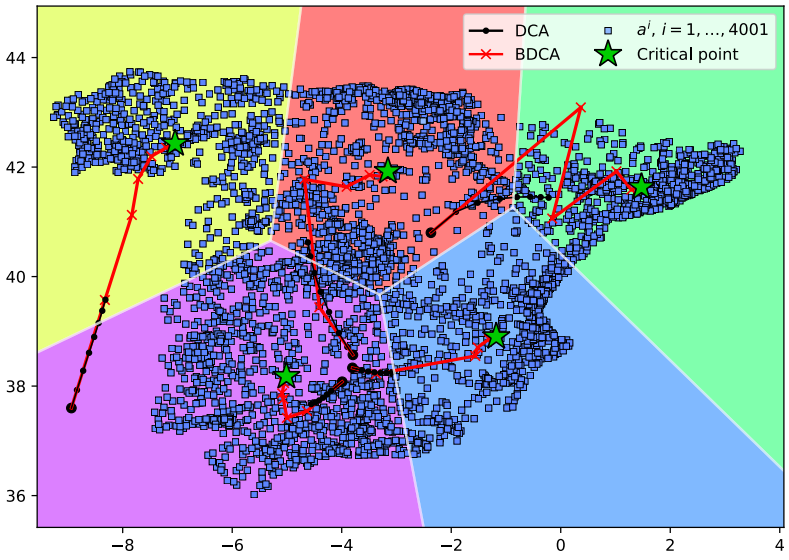


FIG. 5. *Seven iterations of DCA and BDCA are computed from the same starting point for grouping the Spanish cities in the peninsula into five clusters.*

and BDCA to the clustering problem (5.1) *from a random starting point (composed by a quintet of points in* $\mathbb{R}^2$*), with the parameters* $\rho = \frac{1}{10}$, $\alpha = 0.1$, $\beta = 0.5$, *and* $\overline{\lambda}_1 = 5$. *Both algorithms converge to the same critical point, but it is apparent that the line search of BDCA makes it faster.*

Let us demonstrate that the behavior shown in Figure 5 *is not atypical. To do so, let us consider the same problem of the Spanish cities for a different number of clusters* $k \in \{5, 10, 15, 20, 25, 50, 75, 100\}$. *For each of these values, we run BDCA for* 100 *random starting points with coordinates in* $]-9.26, 3.27[\times]36.02, 43.74[$ *(the range of the geographical coordinates of the cities). The algorithm was stopped when the relative error of the objective function* ϕ *was smaller than* 10^{-3}. *Then, DCA was run from the same starting point until the same value of the objective function was reached, which did not happen in* 31 *instances because DCA* failed *(by which we mean that it converged to a worse critical point). In Figure* 6 *we have plotted the ratios between the running time and the number of iterations, except for those instances where DCA failed. On average, BDCA was* 16 *times faster than DCA, and DCA needed* 18 *times more iterations to reach the same objective value as BDCA.*

EXPERIMENT 5.2 (clustering random points in an m-dimensional box). *In this numerical experiment, we generated* n *random points in* $\mathbb{R}^m$ *whose coordinates were drawn from a normal distribution having a mean of* 0 *and a standard deviation of* 10, *with* $n \in \{500, 1000, 5000, 10{,}000\}$ *and* $m \in \{2, 5, 10, 20\}$. *For each pair of values of* n *and* m, 10 *random starting points were chosen and BDCA was run to solve the* k*-clustering problem until the relative error of the objective function was smaller than* 10^{-3}, *with* $k \in \{5, 10, 15, 20, 25, 50, 75, 100\}$. *As in Experiment* 5.1, *we run DCA from the same starting point as BDCA until the same value of the objective function was reached. The DCA failed to do so in* 123 *instances. The ratios between the respective running times are shown in Figure* 7. *On average, BDCA was* 13.7 *times faster than DCA.*

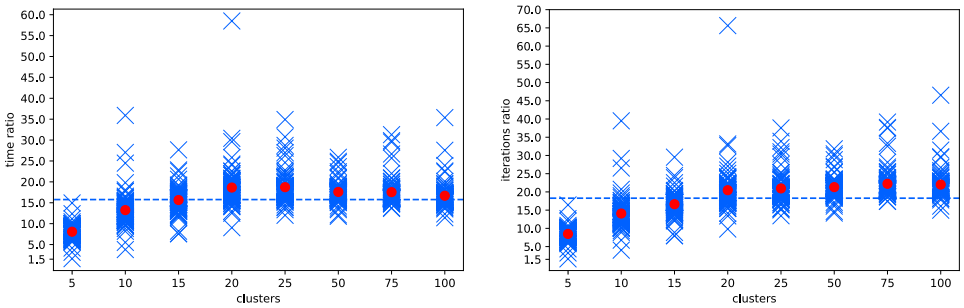
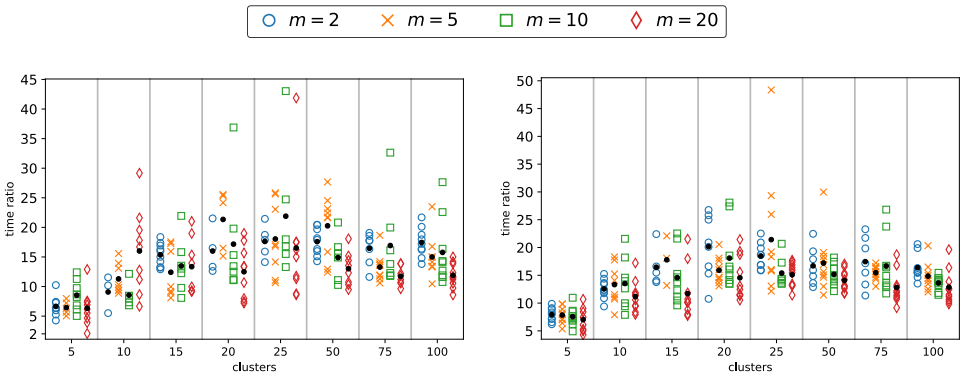
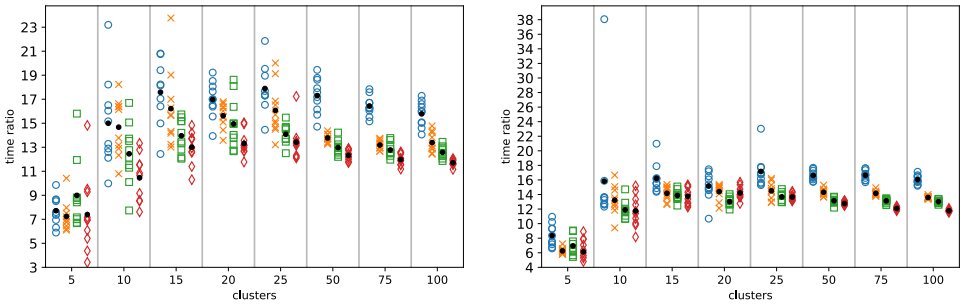


FIG. 6. *Comparison between DCA and BDCA for solving the clustering problem of the cities in the Spanish peninsula described in Experiment* 5.1. *We represent the ratios of running time (left) and number of iterations (right) between DCA and BDCA for* 100 *random instances for different values of the number of clusters* $k \in \{5, 10, 15, 20, 25, 50, 75, 100\}$. *The dashed line shows the overall average ratio, and the red dots represent the average ratio for each value of* k.



(a) $n = 500$ (b) $n = 1000$



(c) $n = 5000$ (d) $n = 10{,}000$

FIG. 7. *Comparison between DCA and BDCA for solving the clustering problems with random data described in Experiment* 5.2. *For each value of* $n \in \{500, 1000, 5000, 10{,}000\}$ *and* $m \in \{2, 5, 10, 20\}$ *we represent the ratios of running time between DCA and BDCA for* 10 *random starting points for different values of the number of clusters* $k \in \{5, 10, 15, 20, 25, 50, 75, 100\}$. *The black dots represent the average ratios.*

5.2. The multidimensional scaling problem. Given only a table of distances between some objects, known as the *dissimilarity matrix*, MDS is a technique that permits us to represent the data in a small number of dimensions (usually two or three). If the objects are defined by n points $x^1, x^2, \ldots, x^n$ in $\mathbb{R}^q$, the entries δ_{ij} of the dissimilarity matrix can be defined by the Euclidean distance between these points,

$$\delta_{ij} = \|x^i - x^j\| := \mathrm{d}_{ij}(X),$$

where we denote by X the $n \times q$ matrix whose rows are $x^1, x^2, \ldots, x^n$.

Given a target dimension $p \leq q$, the metric MDS problem consists in finding n points in $\mathbb{R}^p$, which are represented by an $n \times p$ matrix X^*, such that the quantity

$$Stress(X^*) := \sum_{i<j} w_{ij} \left(\mathrm{d}_{ij}(X^*) - \delta_{ij}\right)^2$$

is smallest, where w_{ij} are nonnegative weights. As shown in [18, p. 236], this problem can be equivalently reformulated as a DC problem of type (1.1) by setting

$$g(X) := \frac{1}{2} \sum_{i<j} w_{ij} \mathrm{d}_{ij}^2(X) + \frac{\rho}{2} \|X\|^2,$$
$$h(X) := \sum_{i<j} w_{ij} \delta_{ij} \mathrm{d}_{ij}(X) + \frac{\rho}{2} \|X\|^2,$$

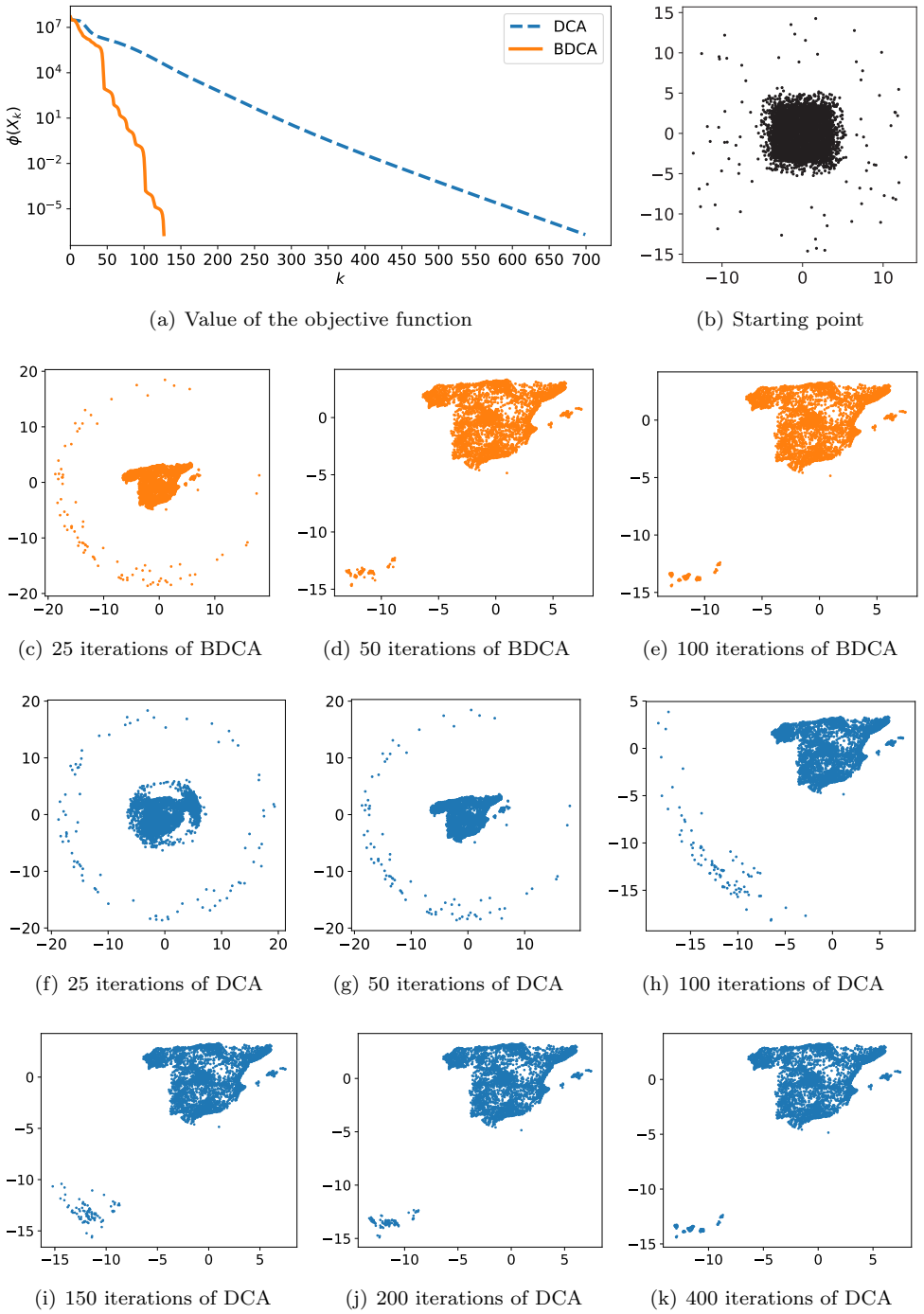
for some $\rho \geq 0$. Moreover, it is clear that g is differentiable while h is not. However, the subgradient of h can be explicitly computed; see [18, section 4.2]. Both functions are strongly convex for any $\rho > 0$.

For this problem we replicated some of the numerical experiments in [18], where the authors demonstrate the good performance of DCA for solving MDS problems. Our main aim here is showing that even for those problems where DCA works well in practice, BDCA is able to outperform it.

In our experiments, we set the weights $w_{ij} = 1$ and the starting points were generated as in [18]. First, we randomly chose a matrix $\widetilde{X}_0 \in \mathbb{R}^{n \times p}$ with entries in $]0, 10[$. Then, the starting point was set as $X_0 := \left(I - (1/n)ee^T\right) \widetilde{X}_0$, where I and e denote the identity matrix and the vector of ones in $\mathbb{R}^n$, respectively. We used the parameters $\rho = \frac{1}{np}$, $\alpha = 0.05$, $\overline{\lambda}_1 = 3$, and $\beta = 0.1$.

EXPERIMENT 5.3 (MDS for Spanish cities). *Consider the dissimilarity matrix defined by the distances between the* 4155 *Spanish cities with more than* 500 *residents, including this time those outside the peninsula to make the problem more difficult. The optimal value of this MDS problem is zero. In Figure* 8(b) *we have represented a starting point of the type* $X_0 := \left(I - (1/4155)ee^T\right) \widetilde{X}_0$*, where* $\widetilde{X}_0 \in \mathbb{R}^{4155 \times 2}$ *was randomly chosen with entries in* $]0, 10[$*. In Figure* 8(c)–(k) *we plot the iterations of DCA and BDCA. As shown in Figure* 8(a)*, although both DCA and BDCA converged to the optimal solution, DCA required five times more iterations than BDCA to reach the same accuracy.*

To demonstrate that the advantage shown in Figure 8 *is not unusual, we ran both algorithms from* 100 *different random starting points either until the value of the objective function was smaller than* 10^{-6} *or until* $\phi(X_k) - \phi(X_{k+1}) < 10^{-6}$*. This second stopping criterion was used in* 32 *instances, and in all of them the value of* ϕ *was approximately equal to* 26683.66*. The running time and the number of iterations of both algorithms are plotted in Figure* 9*. On average, BDCA was* 3.9 *times faster*



(a) Value of the objective function

(b) Starting point

(c) 25 iterations of BDCA

(d) 50 iterations of BDCA

(e) 100 iterations of BDCA

(f) 25 iterations of DCA

(g) 50 iterations of DCA

(h) 100 iterations of DCA

(i) 150 iterations of DCA

(j) 200 iterations of DCA

(k) 400 iterations of DCA

FIG. 8. *Comparison between DCA and BDCA when they are applied to the MDS problem of the Spanish cities described in Experiment* 5.3 *from the same random starting point.*

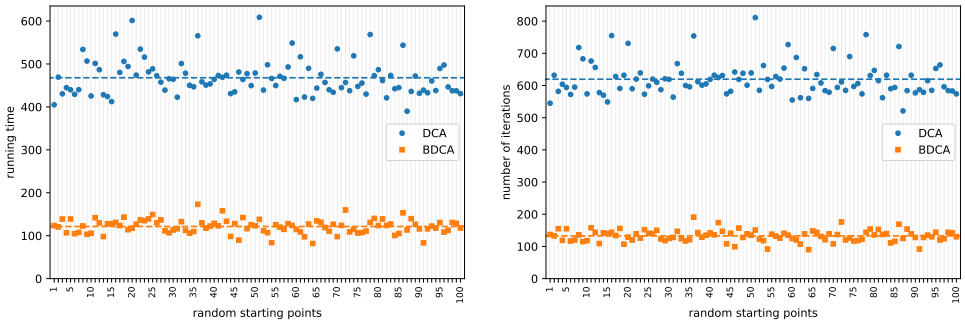


FIG. 9. *Comparison between DCA and BDCA for solving the MDS problem of the Spanish cities described in Experiment* 5.3. *We represent the running time (left) and number of iterations (right) of DCA and BDCA for* 100 *random instances. The dashed lines show the averages.*

than DCA. Further, BDCA was always more than 2.9 *times faster than DCA, and the number of iterations required by DCA was always more than* 3.5 *times higher (on average, it was* 4.7 *times higher). In fact, the minimum time required by DCA within all the random instances (*389.9 *seconds) was* 2.2 *times higher than the maximum time spent by BDCA (*173.2 *seconds).*

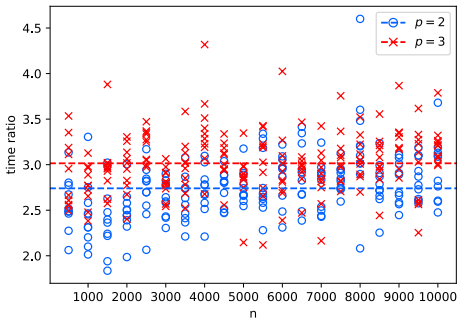
EXPERIMENT 5.4 (MDS with random data). *To test randomly generated data, we considered two cases:*

- *Case* 1*: The dissimilarities are distances between objects in* $\mathbb{R}^p$*; thus, the optimal value is* 0.
- *Case* 2*: The dissimilarities are distances between objects in* $\mathbb{R}^{2p}$*; hence, the optimal value is unknown a priori.*

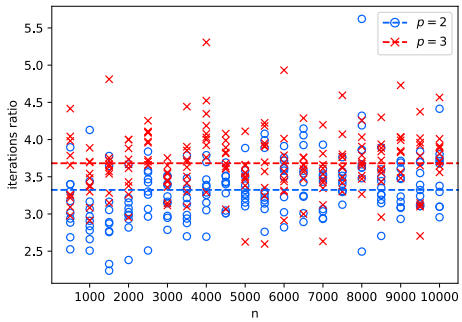
The data was obtained by generating a matrix M *in* $\mathbb{R}^{n\times p}$ *and* $\mathbb{R}^{n\times 2p}$ *with entries randomly drawn from a normal distribution having a mean of* 0 *and a standard deviation of* 10*. Then, the values of* δ_{ij} *were determined by the distance matrix between the rows of* M*. We used the same stopping criteria as in* [18]*: for Case* 1*, the algorithms were stopped when the value of the merit function was smaller than* 10^{-6}*, while for Case* 2*, they were stopped when the relative error of the objective function was smaller than* 10^{-3}*.*

The ratios between the respective running times and number of iterations of DCA and BDCA are shown in Figure 10*. On average, BDCA was* 2.6 *times faster than DCA, and the advantage was bigger both for Case* 1 *and for* $p = 3$*. For Case* 2 *we can find some instances where BDCA was only* 1.5 *times faster than DCA. In Figure* 11 *we observe that these instances seem to be outliers, for which DCA was faster than usual. The value of the objective function with respect to time of both algorithms for a particular large random instance is plotted in Figure* 12*.*

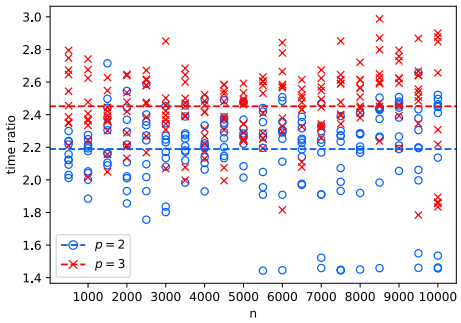
6. Concluding remarks. We have developed a version of the boosted DC algorithm proposed in [2] for solving DC programming problems when the objective function is not differentiable. Our convergence results were obtained under some standard assumptions. The global convergence and convergence rate were established assuming the strong Kurdyka–Łojasiewicz inequality. It remains as an open question whether the results still hold under the Kurdyka–Łojasiewicz inequality, i.e., the corresponding inequality associated with the limiting subdifferential instead of the Clarke subdifferential. This is a topic for future research.



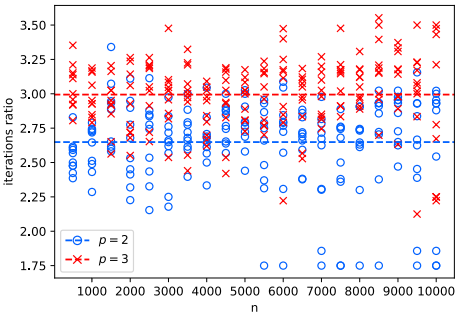
(a) Case 1 (running time ratio)



(b) Case 1 (number of iterations ratio)

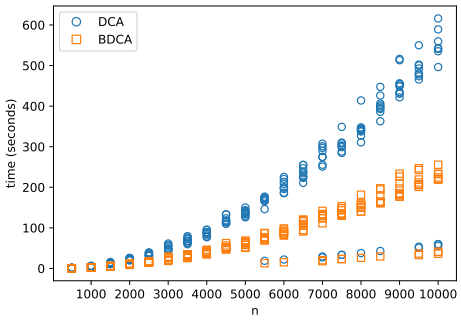


(c) Case 2 (running time ratio)

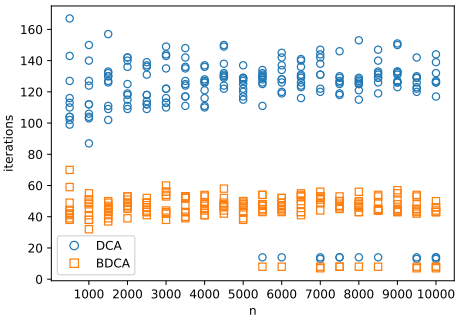


(d) Case 2 (number of iterations ratio)

FIG. 10. *Comparison between DCA and BDCA for solving the MDS problems with random data described in Experiment* 5.4. *We represent the ratios of running time and number of iterations between DCA and BDCA for* 10 *random instances for each value of* $n \in \{500, 1000, \ldots, 10{,}000\}$ *and* $p \in \{2, 3\}$. *For each* p, *the average value is represented with a dashed line.*



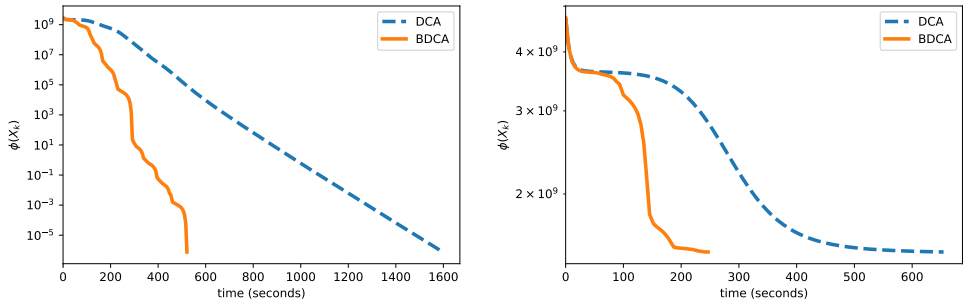
(a) Running time



(b) Number of iterations

FIG. 11. *Running time and number of iterations for DCA and BDCA when applied to the random data described in Experiment* 5.4 *for Case* 2 *with* $p = 2$.

We have applied our algorithm for solving two important problems in data science, namely, the minimum sum-of-squares clustering problem and the multidimensional scaling problem. Our numerical experiments indicate that BDCA outperforms DCA, being on average more than 16 times faster in the first problem and nearly 3 times



(a) Case 1 (b) Case 2

FIG. 12. *Value of the objective function of DCA and BDCA (using logarithmic scale) against CPU time for one particular random instance of each of the two test cases in Experiment* 5.4 *(with* $p = 3$ *and* $n = 10{,}000$*).*

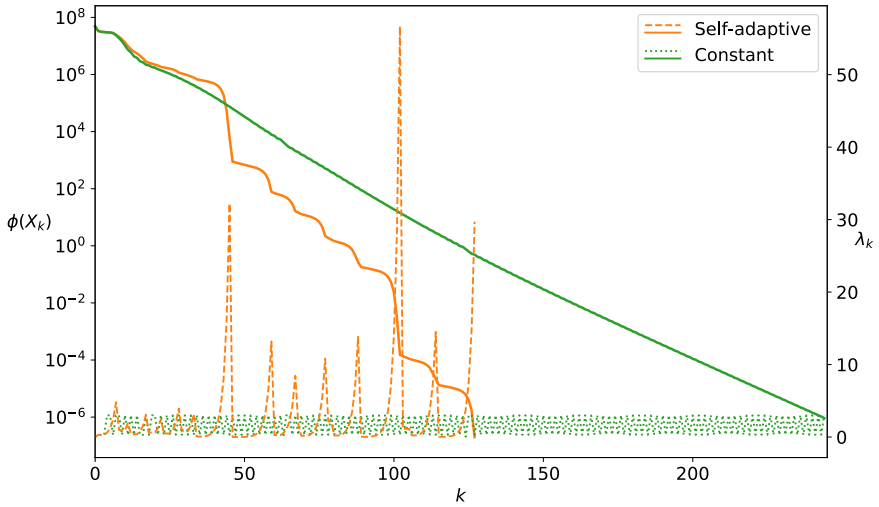


FIG. 13. *Comparison of the self-adaptive and the constant (with* $\overline{\lambda}_k = 3$*) choices for the trial step sizes of BDCA in step* 4*, using the same starting point as in Figure* 8*. The plot includes two scales, a logarithmic one for the objective function values and another one for the step sizes (which are represented with discontinuous lines).*

faster in the second problem, in both computational time and number of iterations. In general, the advantage of BDCA against DCA will always depend on two key factors: the difficulty in solving the subproblems $(\mathcal{P}_k)$ and the number of backtracking steps needed at each iteration. A relatively small backtracking parameter $\beta \approx 0.1$ seems to work well in practice.

An important novelty of the proposed algorithm is the flexibility in the choice of the trial step size $\overline{\lambda}_k$ in the line search step of BDCA, which had to be constant in our previous work [2]. A comparison of both strategies is shown in Figure 13 using the same starting point as in Figure 8, where we can observe that each drop in the function value of the self-adaptive strategy was originated by a large increase of the step size. Although BDCA with constant choice was slower, it still needed three times fewer iterations than DCA; see Figure 8(a). The complete freedom in the choice of $\overline{\lambda}_k$

permits us to use the information available from previous iterations, as done in section 5 with what we call the *self-adaptive trial step size*. Roughly, this strategy allowed us to obtain a two times speed up of BDCA in all our numerical experiments, when compared with the constant strategy. There are many possibilities in the choice of the trial step size to investigate, which could further improve the performance of BDCA.

Finally, we would like to mention that applications of BDCA to the bilevel hierarchical clustering problem [27] and the multicast network design problem [15] can also be considered. However, due to the inclusion of a penalty and a smoothing parameter, the DC objective function associated with these problems changes at each iteration; see [15, 27] for details. Therefore, the applicability of BDCA should be justified in this setting. This serves as an interesting question for future research.

Acknowledgments. The authors wish to thank Aris Daniilidis for his help with Remark 4.2 and Boris Mordukhovich for his suggestion of using the symmetric subdifferential instead of Clarke's, as mentioned in Remark 4.4. The authors are also grateful to the associate editor, Prof. Hedy Attouch, and the referees for their constructive comments, which helped improve the paper significantly.

REFERENCES

[1] N. T. An and N. M. Nam, *Convergence analysis of a proximal point algorithm for minimizing differences of functions*, Optimization, 66 (2017), pp. 129–147.
[2] F. J. Aragón Artacho, R. Fleming, and P. T. Vuong, *Accelerating the DC algorithm for smooth functions*, Math. Program., 169B (2018), pp. 95–118.
[3] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, *Proximal alternating minimization and projection methods for nonconvex problems. An approach based on the Kurdyka–Łojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457.
[4] H. Attouch and J. Bolte, *On the convergence of the proximal algorithm for nonsmooth functions involving analytic features*, Math. Program., 116 (2009), pp. 5–16.
[5] H. Attouch, J. Bolte, and B. F. Svaiter *Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods*, Math. Program., 137 (2013), pp. 91–129.
[6] S. Banert and R. Boţ, *A general double-proximal gradient algorithm for d.c. programming*, Math. Program., 178 (2019), pp. 301–326.
[7] H. H. Bock, *Clustering and neural networks*, in Advances in Data Science and Classification, Springer, Berlin, 1998, pp. 265–277.
[8] J. Bolte, A. Daniilidis, and A. Lewis, *The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems*, SIAM J. Optim., 17 (2007), pp. 1205–1223.
[9] J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota, *Clarke subgradients of stratifiable functions*, SIAM J. Optim., 18 (2007), pp. 556–572.
[10] J. Bolte, A. Daniilidis, O. Ley, and L. Mazet, *Characterizations of Lojasiewicz inequalities: Subgradient flows, talweg, convexity*, Trans. Amer. Math. Soc., 362 (2010), pp. 3319–3363.
[11] J. Bolte, S. Sabach, and M. Teboulle, *Proximal alternating linearized minimization for nonconvex and nonsmooth problems*, Math. Program., 146 (2013), pp. 459–494.
[12] F. H. Clarke, *Optimization and Nonsmooth Analysis*, 2nd ed., Classics in Appl. Math. 5, SIAM, Philadelphia, 1990.
[13] T. H. Cuong, N. D. Yen, and Y. C. Yao, *Qualitative Properties of the Minimum Sum-of-Squares Clustering Problem*, arXiv:1810.02057, 2018.
[14] M. Fukushima and H. Mine, *A generalized proximal point algorithm for certain non-convex minimization problems*, Internat. J. Systems Sci., 12 (1981), pp. 989–1000.
[15] W. Geremew, N. M. Nam, A. Semenov, V. Boginski, and E. Pasiliao, *A DC programming approach for solving multicast network design problems via the Nesterov smoothing technique*, J. Global Optim., 72 (2018), pp. 705–729.
[16] K. Kurdyka, *On gradients of functions definable in o-minimal structures*, Ann. Inst. Fourier (Grenoble), 48 (1998), pp. 769–783.
[17] H. A. Le Thi, V. N. Huynh, and T. Pham Dinh, *Convergence analysis of difference-of-convex algorithm with subanalytic data*, J. Optim. Theory Appl., 179 (2018), pp. 103–126.

[18] H. A. Le Thi and T. Pham Dinh, *D.C. programing approach to the multidimensional scaling problem*, in From Local to Global Optimization, P. Pardalos and P. Varbrand, eds, Kluwer, Dordrecht, 2001, pp. 231–276.
[19] H. A. Le Thi and T. Pham Dinh, *The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems*, Ann. Oper. Res., 133 (2005), pp. 23–46.
[20] H. A. Le Thi and T. Pham Dinh, *DC programming and DCA: Thirty years of developments*, Math. Program., 169 (2018), pp. 5–68.
[21] H. A. Le Thi and T. Pham Dinh, *Recent advances in DC programming and DCA*, in Transactions on Computational Intelligence, N. T. Nguyen and H. A. Le Thi, eds., Lecture Notes in Comput. Sci. 8342, Springer, Berlin, 2014, pp. 1–37.
[22] H. A. Le Thi, T. Pham Dinh, and L. D. Muu, *Numerical solution for optimization over the efficient set by D.C. optimization algorithms*, Oper. Res. Lett., 19 (1996), pp. 117–128.
[23] S. Łojasiewicz, *Ensembles semi-analytiques*, Institut des Hautes Etudes Scientifiques, Bures-sur-Yvette (Seine-et-Oise), France, 1965.
[24] H. Mine and M. Fukushima, *A minimization method for the sum of a convex function and a continuously differentiable function*, J. Optim. Theory Appl., 33 (1981), pp. 9–23.
[25] B. S. Mordukhovich, *Variational Analysis and Applications*, Springer Monogr. Math., Springer, New York, 2018.
[26] A. Moudafi and P. Mainge, *On the convergence of an approximate proximal method for DC functions*, J. Comput. Math., 24 (2006), pp. 475–480.
[27] N. M. Nam, W. Geremew, R. Reynolds, and T. Tran, *Nesterov's smoothing technique and minimizing differences of convex functions for hierarchical clustering*, Optim. Lett., 12 (2018), pp. 455–473.
[28] D. Noll, *Convergence of non-smooth descent methods using the Kurdyka–Łojasiewicz inequality*, J. Optim. Theory Appl., 160 (2014), pp. 553–572.
[29] W. de Oliveira and M. P. Tcheou, *An inertial algorithm for DC programming*, Set-Valued Var. Anal., 27 (2019), pp. 895–919.
[30] B. Ordin and A. M. Bagirov, *A heuristic algorithm for solving the minimum sum-of-squares clustering problems*, J. Global Optim., 61 (2015), pp. 341–361.
[31] T. Pham Dinh and H. A. Le Thi, *A DC optimization algorithm for solving the trust-region subproblem*, SIAM J. Optim., 8 (1998), pp. 476–505.
[32] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NY, 1972.
[33] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Grundlehren Math. Wiss. 317, Springer, New York, 1998.
[34] P. D. Tao and H. A. Le Thi, *Convex analysis approach to DC programming: Theory, algorithms and applications*, Acta Math. Vietnam., 22 (1997), pp. 289–355.
[35] J. F. Toland, *On subdifferential calculus and duality in non-convex optimization*, Bull. Soc. Math. Fr. Mém., 60 (1979), pp. 177–183.
[36] H. M. Xu, H. Xue, X. H. Chen, and Y. Y. Wang, *Solving indefinite kernel support vector machine with difference of convex functions programming*, in Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.